UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/847,535	05/01/2001	Barry Bond	MS1-0665US	4017

22801 7590 07/02/2008

LEE & HAYES PLLC
421 W RIVERSIDE AVENUE SUITE 500
SPOKANE, WA 99201

EXAMINER	
STEVENS, THOMAS H	

ART UNIT	PAPER NUMBER
2121	

MAIL DATE	DELIVERY MODE
07/02/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	09/847,535	BOND ET AL.	
	Examiner	Art Unit	
	THOMAS H. STEVENS	2121	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *05 June 2008*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-26,34-42,45 and 46* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-26,34-42,45 and 46* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *05/07/2008 & 06/05/2008*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-26,34-42,45 and 46 were examined.

Section I: Non-Final Rejection

RCE

2. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on 05/07/2008 has been entered.

Allowable Subject Matter

3. The indicated allowability of claims 1-26,34-42,45 and 46 is withdrawn in view of the newly discovered reference(s) to Parry. Rejections based on the newly cited reference(s) follow.

Duplicate Claims Warning

4. Applicants are advised that should claim 38 be found allowable, claim 39 will be objected to under 37 CFR 1.75 as being a substantial duplicate thereof. When two claims in an application are duplicates or else are so close in content that they both cover the same thing, despite a slight difference in wording, it is proper after allowing

one claim to object to the other as being a substantial duplicate of the allowed claim. See MPEP § 706.03(k).

Claim Rejections - 35 USC § 102

5. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

6. Claims 1-26,34-38,40-42,45 and 46 rejected under 35 U.S.C. 102(e) as being anticipated by Parry (US 6,845,508; Parry). Parry discloses a class driver for use in a computer operating system (abstract).

The applied reference has a common assignee with the instant application. Based upon the earlier effective U.S. filing date of the reference, it constitutes prior art under 35 U.S.C. 102(e). This rejection under 35 U.S.C. 102(e) might be overcome either by a showing under 37 CFR 1.132 that any invention disclosed but not claimed in the reference was derived from the inventor of this application and is thus not the invention "by another," or by an appropriate showing under 37 CFR 1.131.

Claim 1. One or more computer-readable media (column 3, lines 40-43 and column 4, line 59)having stored thereon computer-executable instructions (API, column 3, lines

40-43)of implementing a kernel (column 4, lines 8-12) emulator (column 3, line 53-55)for non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)program modules that, when executed by one or more processors, causes the one or more processors to perform actions comprising: intercepting non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)kernel (column 4, lines 8-12) calls from non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)program modules, the non- native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12) calls calling a native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12) having access to hardware through one or more device drivers and hardware interfaces native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)to the native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12); converting the intercepted non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)kernel (column 4, lines 8-12) calls into native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column

4, lines 17-21)kernel (column 4, lines 8-12) calls; and delivering the converted native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12) calls to the native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12) without the non- native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)program modules being modified to target a native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)platform running the native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12) on which the non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e., column 4, lines 38-52)program modules are not designed to run, thereby facilitating interoperability of the non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)program modules within the native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)platform.

Claim 2. One or more computer-readable media (column 3, lines 40-43 and column 4, line 59)as recited in claim 1, wherein the converting further comprises translating (same as cross-platforming, column 3, lines 32-36; e.g., Windows and Windows NT, column 4, lines 30-51) a non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)paradigm for passing parameters into a native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)paradigm for passing parameters.

Claim 3. One or more computer-readable media (column 3, lines 40-43 and column 4, line 59)as recited in claim 1, wherein the converting further comprises translating (same as cross-platforming, column 3, lines 32-36; e.g., Windows and Windows NT, column 4, lines 30-51) non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)CPU instructions into native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)CPU instructions.

Claim 4. One or more computer-readable media (column 3, lines 40-43 and column 4, line 59)as recited in claim 1, wherein the converting further comprises translating (same as cross-platforming, column 3, lines 32-36; e.g., Windows and Windows NT, column 4, lines 30-51) addresses from non-native (example of a 32 bit driver that is design to

provide common architecture to Windows i.e. Column 4, lines 38-52) length into native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21) length.

Claim 5. One or more computer-readable media (column 3, lines 40-43 and column 4, line 59)as recited in claim 1, wherein the converting further comprises converting non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52) argument format into native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21) argument format.

Claim 6. One or more computer-readable media (column 3, lines 40-43 and column 4, line 59)as recited in claim I, wherein the converting further comprises translating (same as cross-platforming, column 3, lines 32-36; e.g., Windows and Windows NT, column 4, lines 30-51) words from non- native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)word size into native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21) word size.

Claim 7. One or more computer-readable media (column 3, lines 40-43 and column 4, line 59)as recited in claim 1, wherein the kernel (column 4, lines 8-12) emulator (column

3, line 53-55)further comprises limiting addressable memory (abstract)to a range addressable by non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52) program modules.

Claim 8. One or more computer-readable media (column 3, lines 40-43 and column 4, line 59)as recited in claim 1, wherein the kernel (column 4, lines 8-12) emulator (column 3, line 53-55)further comprises managing memory (abstract)space that is accessible to both native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)and non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52) program modules.

Claim 9. One or more computer-readable media (column 3, lines 40-43 and column 4, line 59)as recited in claim 1, wherein the kernel (column 4, lines 8-12) emulator (column 3, line 53-55)further comprises synchronizing a native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21) shared data structure (data structures that drivers use to communicate with each other, column 4, lines 1-3) with a non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52) shared data structure (data structures that drivers use to communicate with each other, column 4, lines 1-3).

Claim 10. One or more computer-readable media (column 3, lines 40-43 and column 4, line 59)as recited in claim 1, wherein the kernel (column 4, lines 8-12) emulator (column 3, line 53-55)further comprises: managing memory (abstract)space accessible to both native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)and non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52) program modules; and mapping (column 8, lines 434-46, mapping of buffers) versions of process shared data structures (data structures that drivers use to communicate with each other, column 4, lines 1-3) (process SDSs) and versions of thread shared data structures (data structures that drivers use to communicate with each other, column 4, lines 1-3) (thread SDSs) between native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)and the non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)program modules.

Claim 11. An operating system on the one or more computer-readable media, comprising: a native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12) configured to receive calls from native (example of a 32 bit driver that is design to provide common architecture to Windows

i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)program modules; and a kernel (column 4, lines 8-12) emulator (column 3, line 53-55)as recited in claim 1 configured to receive and convert calls from non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)program modules for direct handling by the native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12) without the non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)program modules being modified to natively call the native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12).

Claim 12. An operating system on the one or more computer-readable media, comprising: a native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12) configured to receive calls from native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)APIs; a kernel (column 4, lines 8-12) emulator (column 3, line 53-55)as recited in claim 1 configured to receive calls from non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)APIs for direct

execution by the native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)APIs without the non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)APIs being modified to natively utilize the native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)APIs.

Claim 13. A method of emulating a kernel (column 4, lines 8-12) for non-native program modules, the method comprising: intercepting non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)kernel (column 4, lines 8-12) calls from non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)program modules, the non- native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12) calls calling a native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12) having access to hardware through one or more device drivers and hardware interfaces native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)to the native (example of a 32 bit driver that is design to provide

common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12); converting the intercepted non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)kernel (column 4, lines 8-12) calls into native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12) calls; and delivering the converted native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12) calls to the native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12) without the non- native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)program modules being modified to target native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)platform running the native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12) on which the non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)program modules are not designed to run.

Claim 14. A method as recited in claim 13, wherein the converting step comprises translating (same as cross-platforming, column 3, lines 32-36; e.g., Windows and Windows NT, column 4, lines 30-51) a non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)paradigm for passing parameters into a native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)paradigm for passing parameters.

Claim 15. A method as recited in claim 13, wherein the converting step comprises translating (same as cross-platforming, column 3, lines 32-36; e.g., Windows and Windows NT, column 4, lines 30-51) non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)CPU instructions into native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)CPU instructions.

Claim 16. A method as recited in claim i3, wherein the converting step comprises translating (same as cross-platforming, column 3, lines 32-36; e.g., Windows and Windows NT, column 4, lines 30-51) addresses from non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)length into native (example of a 32 bit driver that is design to provide common

architecture to Windows i.e. Column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)length.

Claim 17. A method as recited in claim 13, wherein the converting step comprises translating (same as cross-platforming, column 3, lines 32-36; e.g., Windows and Windows NT, column 4, lines 30-51) words from non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)word size into native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)word size.

Claim 18. A method as recited in claim 13 further comprising limiting addressable memory (abstract)to a range addressable by non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)program modules.

Claim 19.A method as recited in claim 13 further comprising synchronizing a native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)shared data structure (data structures that drivers use to communicate with each other, column 4, lines 1-3) with a non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)shared data structure

(data structures that drivers use to communicate with each other, column 4, lines 1-3).

Claim 20, A method as recited in claim i3 further comprising mapping (column 8, lines 434-46, mapping of buffers) versions of process shared data structures (data structures that drivers use to communicate with each other, column 4, lines 1-3) (SDSs) between native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)and non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)program modules.

Claim 21. A method as recited in claim 20, wherein a process SDS of a native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)program module includes a pointer to a process SDS of a non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)program module.

Claim 22. A method as recited in claim 20, wherein a process SDS of a non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)program module includes a pointer to a process SDS of a native (example of a 32 bit driver that is design to provide common

architecture to Windows i.e. Column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)program module.

Claim 23. A method as recited in claim 13 further comprising mapping (column 8, lines 434-46, mapping of buffers) versions of thread shared data structures (data structures that drivers use to communicate with each other, column 4, lines 1-3) (SDSs) data structure between native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)and non- native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)program modules.

Claim 24. A method as recited in claim 23, wherein a thread SDS of a native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)program module includes a pointer to a thread SDS of a non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)program module.

Claim 25. A method as recited in claim 23, wherein a thread SDS of a non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)program module includes a pointer to a thread

SDS of a native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)program
module.

Claim 26. A computer comprising: one or more processors; and memory (abstract)coupled to the one or more processors, the memory (abstract)storing thereon computer-executable instructions (API, column 3, lines 40-43)that, when executed by the one or more processors, perform a method comprising: intercepting non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)kernel (column 4, lines 8-12) calls from non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)program modules, the non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)kernel (column 4, lines 8-12) calls calling a native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12) having access to hardware through one or more device drivers and hardware interfaces native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)to the native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-

21)kernel (column 4, lines 8-12); converting the intercepted non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)kernel (column 4, lines 8-12) calls into native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12) calls; and delivering the converted native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12) calls to the native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12) without the non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)program modules being modified to target native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)platform running the native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12) on which the non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)program modules are not designed to run the method as recited in claim 13.

Claim 34. A method comprising: emulating a non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)kernel (column 4, lines 8-12) for a native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)computing platform by converting non- native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12) calls calling a native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12) from non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)applications into native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12) calls to the native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12), without the non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)applications being modified to target the native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)computing platform on which the non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-

52)applications are not designed to run.

Claim 35. A method as recited in claim 34, wherein the emulating step further comprises: translating (same as cross-platforming, column 3, lines 32-36; e.g., Windows and Windows NT, column 4, lines 30-51) non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)CPU instructions into native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)CPU instructions; translating (same as cross-platforming, column 3, lines 32-36; e.g., Windows and Windows NT, column 4, lines 30-51) addresses from non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)length into native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)length; limiting addressable memory (abstract)to a range addressable by non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)program modules.

Claim 36. A method as recited in claim 35, wherein the emulating step further comprises translating (same as cross-platforming, column 3, lines 32-36; e.g., Windows and Windows NT, column 4, lines 30-51) a non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines

38-52)paradigm for passing parameters into a native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)paradigm for passing parameters.

Claim 37. A method as recited in claim 34, wherein the converting step further comprises translating (same as cross-platforming, column 3, lines 32-36; e.g., Windows and Windows NT, column 4, lines 30-51) words from non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)word size into native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)word size.

Claim 38. (Currently amended) A computer comprising one or more computer- readable media having computer-executable instructions (API, column 3, lines 40-43)that, when executed by the computer, perform a method comprising emulating a non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)kernel (column 4, lines 8-12) for a native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)computing platform by converting non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)kernel (column 4, lines 8-12) calls calling a native (example of a 32 bit driver that is design to provide common architecture to Windows

i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12) from non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)applications into native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12) calls to the native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12) without the non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)applications being modified to target the native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)computing platform on which the non- native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)applications are not designed to run the method as recited in claim 34.

Claim 39. A computer-readable medium having computer-executable instructions that, when executed by a computer, emulates a non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)kernel (column 4, lines 8-12) for a native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus,

column 4, lines 17-21)computing platform by converting non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)kernel (column 4, lines 8-12) calls calling a native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12) from non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)applications into native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12) calls without the non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)applications being modified to target on the native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)computing platform on which the non- native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)applications are not designed to run.

Claim 40. One or more computer-readable media (column 3, lines 40-43 and column 4, line 59)having stored thereon instructions implementing a kernel (column 4, lines 8-12) emulator (column 3, line 53-55)for non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)program modules, the instructions, when executed by a computing device, causing the

computing device to emulate a non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)kernel (column 4, lines 8-12) for a native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)computing platform so that non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)kernel (column 4, lines 8-12) calls that call a native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12) from non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)applications are converted into native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12) calls to the native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12) without the non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)applications being modified to target on the native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)computing platform on which the non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)applications are not designed to run.

Claim 41. One or more computer-readable media (column 3, lines 40-43 and column 4, line 59)having stored thereon instructions implementing the kernel (column 4, lines 8-12) emulator (column 3, line 53-55)recited in claim 40, wherein the instructions of implementing the kernel (column 4, lines 8-12) emulator (column 3, line 53-55)comprises: instructions implementing an instruction-translator (same as cross platforming, column 3, lines 32-37)configured to translate non- native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)CPU instructions into native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)CPU instructions; instructions implementing an address-translator configured to translate (same as cross-platform API, column 3, lines 32-36)addresses from non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)length into native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)length; and instructions implementing a memory (abstract)constrainer configured to limit addressable memory (abstract)to a range addressable by non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)program modules.

Claim 42. One or more computer-readable media (column 3, lines 40-43 and column 4,

line 59)having stored thereon instructions of an operating system that, when executed on a computing device, cause the computing device to implement a plurality of modules, the instructions comprising: instructions of implementing a native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12) configured to receive calls from native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)program modules; instructions of implementing a kernel (column 4, lines 8-12) emulator (column 3, line 53-55)as recited in claim 40 configured to receive calls from non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)program modules.

Claim 45. One or more computer-readable media (column 3, lines 40-43 and column 4, line 59)having stored thereon instructions that, when executed by a computing device, causes the computing device to implement a kernel (column 4, lines 8-12) emulator (column 3, line 53-55)for non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52) program modules, the kernel (column 4, lines 8-12) emulator (column 3, line 53-55) comprising: an interceptor configured to intercept non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)kernel (column

4, lines 8-12) calls that call a native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12) from non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)program modules, the native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12) being software that operates system functions; a call-converter configured to convert the non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)kernel (column 4, lines 8-12) calls intercepted by the interceptor into native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12) calls, wherein the call-converter comprises: an instruction-translator (same as cross platforming, column 3, lines 32-37)configured to translate non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)CPU instructions into native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)CPU instructions; an address-translator configured to translate (same as cross-platform API, column 3, lines 32-36)addresses from non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)length into native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column

4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)length; and an I/O unit configured to deliver converted native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12) calls to the native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12), wherein the call-converter enables the non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)program modules to call the native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12) without the non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)program modules being modified to target platform running the native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)kernel (column 4, lines 8-12) for which the non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52)program modules are not designed.

Claim 46. An operating system on a computer-readable medium, comprising: a native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column

4, lines 17-21)kernel (column 4, lines 8-12) configured to receive calls from native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)program modules; a kernel (column 4, lines 8-12) emulator (column 3, line 53-55)as recited in claim 45 configured to receive calls from non-native (example of a 32 bit driver that is design to provide common architecture to Windows i.e. Column 4, lines 38-52; suggestion of a 64 bit bus, column 4, lines 17-21)program modules.

Section II: Response to Arguments

101

7. Rejection is withdrawn.

Conclusion

8. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Mr. Tom Stevens whose telephone number is 571-272-3715.

If attempts to reach the examiner by telephone are unsuccessful, please contact examiner's supervisor Mr. Albert Decady (571-272-3819). The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov.. Answers to questions regarding access to the Private PAIR system, contact the Electronic Business Center (EBC) (toll-free (866-217-9197)).

/Albert Decady /
Supervisory Patent Examiner
Tech Center 2100

/Thomas H. Stevens/

Examiner, Art Unit 2121